IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant	: Greg Christopher, Jr.	Art Unit	: 2191
Serial No.	: 10/716,916	Examiner	: Qing Chen
Filed	: November 18, 2003	Conf. No.	: 6410
Title	: SOFTWARE INSTALLATION VERIFICATION		

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

BRIEF ON APPEAL

Sir:

This brief on appeal is filed under 37 CFR 41.37, thereby perfecting the notice of appeal which was originally filed on September 16, 2008. The sections required by 37 CFR 41.37 follow.

(1) Real Party in Interest

This application is assigned of record to Adobe Systems Incorporated who is hence the real party in interest.

(2) Related Appeals and Interferences

There are no known related appeals or interferences.

(3) Status of Claims

Claims 1-2, 5-10, 12, 14-23, 25, 26 and 28-32 are pending, with claims 1, 10, 19, and 25 being independent. Claims 1-2, 5-10, 12, 14-23, 25, 26 and 28-32 stand rejected. Claims 1-2, 5-10, 12, 14-23, 25, 26 and 28-32 are appealed herein.

(4) Status of Amendments

Claims 3, 11 and 27 were cancelled (without prejudice) in the Response submitted on August 8, 2008, and this amendment to the claims was entered for purposes of appeal in the Advisory Action dated September 19, 2008. No other claim amendments have been filed after final rejection.

(5) Summary of Claimed Subject Matter

The present claims define systems and techniques relating to software installer verification during software product development. For example, as described in the Specification: "A change-tracking system using the systems and techniques described can be used to verify correct installation of a software product with many resources that are being changed frequently (e.g., daily). Change tracking can be efficiently performed across multiple versions of the software product, such as those created for different operating systems, human languages, and custom install options. A system employing the present techniques can track what should actually be installed on a particular day in a software product's life-cycle, including components that need to be installed in other applications' directories, and quickly call attention to inadvertently introduced errors. This can be of particular value in the latter part of a software product's development life-cycle, when engineers may be struggling to meet a set product release date. [...] The advantages of a system using the present techniques to facilitate verification of an installer for a software product under development can be especially important as the product's changes slow to a stop as a release date nears. At this point in time, most resources have expectations of stability, and the date, size, checksum, and permission attributes of installed files are very close to what is desired in the final product. Because an exact attribute identity for each file that has stabilized can be provided, each file can be easily verified as installed correctly, byte-for-byte and attribute-for-attribute. Hundreds of installations can be performed and verified in the space of time it might otherwise take to verify a single combination."[1]

Independent claim 1 defines a method including: generating a comparison of a current software installation, to a target computer, with a previous software installation, to the same target computer, in a series of two or more software installations during a software product development;[2] creating installation data for a resource, based at least in part on the comparison,

---

[1] *See* Specification at page 3, line 13, to page 4, line 9; emphasis added.
[2] *See e.g.,* Specification at page 4, line 27, to page 5, line 7; at page 5, lines 11-29; at ref. # 100 in FIG. 1; and at ref. #'s 200-222 in FIG. 2.

the resource including attributes including a dynamic attribute and a static attribute, the dynamic attribute is an attribute that should have changed between the previous software installation and the current software installation, the static attribute is an attribute that should remain unchanged between the previous software installation and the current software installation;[3] identifying from the installation data the dynamic attribute that was not changed in the current software installation;[4] and presenting potential problems with the current software installation based on the identified dynamic attribute to facilitate verification of an installer for the software product development.[5]

Independent claim 10 defines a software product tangibly embodied in a machine-readable storage device to perform operations including: generating a comparison of a current software installation, to a target computer, with a previous software installation, to the same target computer, in a series of two or more software installations during a software product development;[6] creating installation data for a resource, based at least in part on the comparison, the resource including attributes including a dynamic attribute and a static attribute, the dynamic attribute is an attribute that should have changed between the previous software installation and the current software installation, the static attribute is an attribute that should remain unchanged between the previous software installation and the current software installation;[7] identifying from the installation data the dynamic attribute that was not changed in the current software installation;[8] and presenting potential problems with the current software installation based on

---

[3] *See e.g.,* Specification at page 5, line 27, to page 6, line 3; at page 7, lines 10-15; at page 8, line 23, to page 9, line 25; at ref. #'s 240 and 250 in FIG. 2; and at ref. #'s 430 and 450 in FIG. 4.

[4] *See e.g.,* Specification at page 5, lines 8-11; at page 5, line 30, to page 6, line 3; at page 7, lines 1-2; at ref. # 110 in FIG. 1; and at ref. # 330 in FIG. 3.

[5] *See e.g.,* Specification at page 5, line 29, to page 6, line 17; at ref. # 120 in FIG. 1; at page 7, lines 4-6; at page 10, lines 2-23; at page 12, lines 1-6; at page 13, lines 5-7; at reg. # 260 in FIG. 2; at ref. # 350 in FIG. 3; and at ref. # 1100 in FIG. 11.

[6] *See e.g.,* Specification at page 4, line 27, to page 5, line 7; at page 5, lines 11-29; at ref. # 100 in FIG. 1; and at ref. #'s 200-222 in FIG. 2.

[7] *See e.g.,* Specification at page 5, line 27, to page 6, line 3; at page 7, lines 10-15; at page 8, line 23, to page 9, line 25; at ref. #'s 240 and 250 in FIG. 2; and at ref. #'s 430 and 450 in FIG. 4.

[8] *See e.g.,* Specification at page 5, lines 8-11; at page 5, line 30, to page 6, line 3; at page 7, lines 1-2; at ref. # 110 in FIG. 1; and at ref. # 330 in FIG. 3.

the identified dynamic attribute to facilitate verification of an installer for the software product development.[9]

Independent claim 19 defines a system that includes: a build controller;[10] an install controller comprising a database including a baseline recording expectations of a dynamic attribute and a static attribute for one or more resources associated with a software installer, the dynamic attribute is an attribute that should have changed between a previous software installation and a current software installation, the static attribute is an attribute that should remain unchanged between the previous software installation and the current software installation;[11] and one or more install slave machines;[12] wherein the build controller automatically triggers the install controller to initiate installer tests as part of a software build process, and the install controller automatically dispatches installation to the one or more install slave machines and collects test results to be presented in a report comprising a baseline-update interface.[13]

Independent claim 25 defines a system that includes: a user interface device;[14] and one or more computers[15] operable to interact with the user interface device and to perform operations comprising: generating a comparison of a current software installation, to a target computer, with a previous software installation, to the same target computer, in a series of two or more software installations during a software product development;[16] creating installation data for a resource, based at least in part on the comparison, the resource including attributes including a dynamic

---

[9] *See e.g.,* Specification at page 5, line 29, to page 6, line 17; at ref. # 120 in FIG. 1; at page 7, lines 4-6; at page 10, lines 2-23; at page 12, lines 1-6; at page 13, lines 5-7; at reg. # 260 in FIG. 2; at ref. # 350 in FIG. 3; and at ref. # 1100 in FIG. 11.

[10] *See e.g.,* Specification at page 11, lines 3-14; and at ref. # 510 in FIG. 5.

[11] *See e.g.,* Specification at page 8, line 9, to page 9, line 25; at page 11, lines 3-29; at ref. #'s 420, 430, 450, 460; and at ref. # 520 in FIG. 5.

[12] *See e.g.,* Specification at page 8, lines 7-9; at page 11, lines 15-16; at ref. # 410 in FIG. 4; and at ref. # 530 in FIG. 5.

[13] *See e.g.,* Specification at page 11, lines 16-29; and at ref. # 540 in FIG. 5.

[14] *See e.g.,* Specification at 10, line 24, to page 11, line 2; at page 11, lines 22-26; at page 12, lines 1-23; at ref. # 440 in FIG. 4; at ref. # 540 in FIG. 5; at ref. # 600 in FIG. 6; at ref. # 700 in FIG. 7; and at ref. # 800 in FIG. 8.

[15] *See e.g.,* Specification at page 5, lines 12-26; at page 8, line 8, to page 10, line 9; at page 11, lines 3-29; at ref. # 200 in FIG. 2; at ref. #'s 410 and 420 in FIG. 4; at ref. #'s 510 and 520 in FIG. 5.

[16] *See e.g.,* Specification at page 4, line 27, to page 5, line 7; at page 5, lines 11-29; at ref. # 100 in FIG. 1; and at ref. #'s 200-222 in FIG. 2.

attribute and a static attribute, the dynamic attribute is an attribute that should have changed between the previous software installation and the current software installation, the static attribute is an attribute that should remain unchanged between the previous software installation and the current software installation;[17] identifying from the installation data the dynamic attribute that was not changed in the current software installation;[18] and presenting potential problems with the current software installation based on the identified dynamic attribute to facilitate verification of an installer for the software product development.[19]

(6) Grounds of Rejection to be Reviewed on Appeal

I. Grounds of Rejection I

Claims 1-3, 5-12, 14-18, and 25-32 stand rejected under 35 U.S.C. § 103(a) as allegedly being upatentable over U.S. Patent No. 6,738,970 issued to Kruger et al. (hereinafter "Kruger") in view of U.S. Patent No. 6,560,776 issued to Breggin et al. (hereinafter "Breggin").

II. Grounds of Rejection II

Claims 19, 20, 22 and 23 stand rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over Breggin in view of Kruger.

III. Grounds of Rejection III

Claim 21 stand rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over Breggin in view of Kruger, and further in view of U.S. Patent Publication No. 2002/0156831 issued to Suorsa et al. (hereinafter "Suorsa").

---

[17] *See e.g.,* Specification at page 5, line 27, to page 6, line 3; at page 7, lines 10-15; at page 8, line 23, to page 9, line 25; at ref. #'s 240 and 250 in FIG. 2; and at ref. #'s 430 and 450 in FIG. 4.

[18] *See e.g.,* Specification at page 5, lines 8-11; at page 5, line 30, to page 6, line 3; at page 7, lines 1-2; at ref. # 110 in FIG. 1; and at ref. # 330 in FIG. 3.

[19] *See e.g.,* Specification at page 5, line 29, to page 6, line 17; at ref. # 120 in FIG. 1; at page 7, lines 4-6; at page 10, lines 2-23; at page 12, lines 1-6; at page 13, lines 5-7; at reg. # 260 in FIG. 2; at ref. # 350 in FIG. 3; and at ref. # 1100 in FIG. 11.

(7) Argument

I. Rejection based on Kruger in view of Breggin: Neither of these references, either alone or in combination, describe the subject matter of claims 1-3, 5-12, 14-18, and 25-32.

A. Current Claim Construction Fails to Consider the Subject Matter As A Whole

Independent claim 1 recites, in part (emphasis added), "generating a comparison of a current software installation, to a target computer, with a previous software installation, to the same target computer, in a series of two or more software installations during a software product development; creating installation data for a resource, based at least in part on the comparison, the resource including attributes including a dynamic attribute and a static attribute, the dynamic attribute is an attribute that should have changed between the previous software installation and the current software installation, the static attribute is an attribute that should remain unchanged between the previous software installation and the current software installation; identifying from the installation data the dynamic attribute that was not changed in the current software installation[.]"

In contrast, Kruger describes a method and apparatus for "identifying changes made to a computer system due to software installation" in order to assist computer administrators in installing software on a large number of computer systems, where such administrators "prefer software to be installed over many computer systems in a consistent manner to make supporting that software easier."[20] Kruger's process involves the following[21]:

> The state of the computer system is recorded before the software is installed and, after the software is installed, the recorded state is compared against the state of

[20] *See* Kruger at Title and col. 1, line 61, to col. 2, line 39.
[21] *See* Kruger at Abstract.

> the computer system. Changes are written into a manifest, which may be combined with any new or changed files and an installation program to produce a package which can be sent to another computer system for installation.

Kruger describes, in detail, the process of recording a "before" state[22], installing software[23], recording an "after" state[24], comparing these two states, performing post processing and generating a manifest.[25]

In the 2-26-2008 Response, it was noted that the portion of Kruger cited by the Examiner cannot be considered as teaching creating installation data for a resource that has a dynamic attribute, which is an attribute that should have changed between successive installations on the same target computer. In response, the Examiner cited again to Column 5, lines 18-29, and stated, "the modification date of a file can be considered to be a 'dynamic attribute' since the modification date of the file should change in order to indicate when the change occur between successive installations. The filename of a file can be considered to be a 'static attribute' since the filename of the file should not change in order to maintain file consistency between successive installation."[26] However, this statement by the Examiner defies common sense, since at this cited stage of the processing in Kruger, no installation (let alone successive installations) has yet occurred, and the system has no information regarding whether any of the filename and modification date information being collected will or will not be impacted by an installation.

The Examiner attempts to counter this point by asserting that, "the claim language does not define any specific relationship between the current software installation and the previous

---

[22] *See* Kruger at col. 4, line 28, to col. 7, line 28.
[23] *See* Kruger at col. 7, lines 29-46.
[24] *See* Kruger at col. 7, line 50, to col. 8, line 11.
[25] *See* Kruger at col. 8, line 12, to col. 12, line 50.
[26] *See* 6-16-2008 Office Action at page 22.

software installation. Nor does the claim language offer any clarification on whether the current software installation and the previous software installation are related to the same software program."[27] This assertion is incorrect. Claim 1 explicitly states, "generating a comparison of a current software installation, to a target computer, with a previous software installation, to the same target computer, in a series of two or more software installations during a software product development;" and then states, "creating installation data for a resource, based at least in part on the comparison, the resource including attributes including a dynamic attribute and a static attribute, the dynamic attribute is an attribute that should have changed between the previous software installation and the current software installation[.]" The plain meaning of this claim language, in light of the Specification, is that the current software installation and the previous software installation relate to the software product being developed. If this were not the case, there would be no reason to compare the two, as recited in the claim.

Moreover, this interpretation is further supported by the remainder of the claim language, which states, "presenting potential problems with the current software installation based on the identified dynamic attribute to facilitate verification of an installer for the software product development."[28] This clearly links the software product development with the current software installation and the previous software installation, through the identified dynamic attribute. If the current software installation and the previous software installation, as claimed, were not related to the same software program being developed, there is no way that a comparison between the two installations to find a dynamic attribute that should have changed, but didn't, could be used to facilitate verification of an installer for the software product development.

---

[27] *See* 09-16-2008 Advisory Action at page 2, "Second" point.
[28] Emphasis added.

Thus, the Examiner has failed to consider the claimed subject matter as a whole when developing the claim construction, and the rejection of claim 1 should be overturned for at least this reason.

Further, the Examiner acknowledges that "Kruger does not disclose software product development"[29] and relies on Breggin for this deficiency of Kruger. However, the cited portion of Breggin (col. 3 line 66 to col. 4 line 6) describes an "installation analysis tool that creates a verification or installation database based on an analysis of the install program before the installation" (col. 3, lines 60-63), and does not disclose a software product development. Regarding this point, the Examiner has further stated:[30]

> "...the install program is created by a builder or installer on a computer that is hereinafter referred to as the build computer. The builder or installer writes a program or script describing how the software and supporting files are to be installed on a target computer.") Note that the install program is created by a builder or installer on a computer (software product development).

However, the install program is not the software product. Rather, it the software used to deliver and install the software product, which was developed previously. Breggin does not address software product development, but rather software installation development.[31]

The Examiner attempts to avoid this issue by citing to MPEP § 2111 as supporting the proposition that, "the claims must be given the broadest reasonable treatment and interpreted as broadly as their terms reasonable allow", and arguing that, "the install program is clearly a software product that is developed by a build computer."[32] However, this interpretation is inconsistent with the claim language itself since claim 1 later recites, "presenting potential

---

[29] *See* 11-28-2007 Office Action at page 5.
[30] *See* 6-16-2008 Office Action at page 21.
[31] *See* Breggin at Title, Abstract, and throughout.
[32] *See* 09-16-2008 Advisory Action at page 3, "Fifth" point.

problems with the current software installation based on the identified dynamic attribute to facilitate verification of an installer for the software product development."[33] Thus, the Examiner appears to be asserting that the install program itself, in Breggin, has its own installer, which would make no sense to those of ordinary skill in the art. Moreover, the Examiner has inappropriately ignored the actual legal standard quoted in MPEP § 2111, which states that the claims must be "given their broadest reasonable interpretation consistent with the specification." In the present case, the Examiner has failed to comply with this requirement on multiple fronts.

First, the Examiner has created a claim construction that is inconsistent with the specification by inappropriately parsing the first element of claim 1 ("generating a comparison of a current software installation, to a target computer, with a previous software installation, to the same target computer, in a series of two or more software installations during a software product development") to pull out the phrase "during a software product development", and then addresses the separated elements in isolation. Such a division is expressly prohibited:[34]

> Finally, when evaluating the scope of a claim, every limitation in the claim must be considered. USPTO personnel may not dissect a claimed invention into discrete elements and then evaluate the elements in isolation. Instead, the claim as a whole must be considered.

In the present case, the Examiner has broken the claim element in two, stated that the "current software installation" of the first part corresponds to software being installed to a computer and that the "software product" of the second part corresponds to an installer program used to install software to a computer. This is inconsistent with the Specification and leads to the untenable

---

[33] Emphasis added.
[34] *See e.g.*, MPEP § 2106(II)(C) (emphasis added), citing to Diamond v. Diehr, 450 U.S. 175, 188-89, 209 USPQ 1, 9 (1981).

position that the claim reads on presenting potential problems with a software installation to facilitate verification of an installer for an installer program. Thus, the claim construction offered by the Examiner illustrates that the Examiner is using improper hindsight reasoning to piece together the prior art and also creates an interpretation of the claim language that is inconsistent with the Specification.

Second, when further addressing the artificially created first part of the first claim element, the Examiner has also created an inconsistent interpretation of what constitutes a "software installation", as claimed. As noted above, the Examiner equates the claimed "current software installation" with software being installed to the master computer, but when addressing the alleged "previous software installation" in Kruger, the Examiner states:[35]

> Examiner would like to point out that Kruger's invention is related to software installation and more specifically, to a method and apparatus that identifies changes made to a computer system (i.e., the master computer) caused by the installation of software. One of ordinary skill in the art would readily comprehend that various other software installations must have occurred in the master computer prior to the recording of the "before" state of the master computer. Note that the recorded "before" and "after" states of the master computer include all of the files in the master computer's file storage. Thus, the "before" state of the master computer includes the files from the previous software installations.

Thus, the Examiner is equating the claimed "previous software installation" with the "before" state of the master computer, which is the result of presumed multiple previous software installations. This interpretation of the claim language is inconsistent with the Specification.

---

[35] *See* 09-16-2008 Advisory Action at page 2, "Second" point.

The Specification, in its entirety, makes clear that comparing a current software installation with a previous software installation necessarily involves comparing modifications to a computing system caused by a current software install with modifications to the computing system caused by a previous software install.[36] In stark contrast, the Examiner has now adopted a claim construction that eliminates the need to know anything about how the alleged previous software installation modified the computing system since all that is needed under the Examiner's interpretation is the state of the computer after the presumed prior installations. In other words, the Examiner has decided that the claim reads on not only comparing two software installations (as recited), but also on comparing two states of a computer that reside on either side of a single software installation. This interpretation is inconsistent with the Specification and is thus improper.

Moreover, this claim construction is also improper because it ignores the explicit recitation that there be "a series of two or more software installations[.]"[37] The fact that there is a series of two or more software installations in the claimed subject matter provides support for the proper interpretation of the second element of claim 1: "creating installation data for a resource, based at least in part on the comparison, the resource including attributes including a dynamic attribute and a static attribute, the dynamic attribute is an attribute that should have changed between the previous software installation and the current software installation [...]"[38] The "should have changed" language clearly indicates that there is an attribute of a resource involved in the current installation that is expected to change in the current installation, such as,

---

[36] *See e.g.,* Specification at page 5, lines 2-10.
[37] Emphasis added.
[38] Emphasis added.

because that attribute changed as a result of the previous installation. Kruger fails to describe this subject matter.

Kruger describes a process of recording the "before" state of the master computer before an installation has occurred, and no information is known (as described in Kruger) about the modifications to the computer system that resulted from the Examiner's presumed previous installations. Thus, none of the attributes of the resources in Kruger, for which data is being collected, can be considered a dynamic attribute, which is "an attribute that should have changed between the previous software installation and the current software installation", as claimed.

In response to this point, the Examiner now further cites to column 1, lines 40-45, and column 6, lines 51-58, of Kruger, and equates the file modification data and the system registry entry with the claimed dynamic attribute, stating, "One of ordinary skill in the art would readily recognize that when the difference calculator detects a change in a file (i.e., modification), the last modification date of the file should be changed to reflect the new date of the change. [...] [And] one of ordinary skill in the art would also readily recognize that whenever a software is installed, the registry information for the software should be changed and reflected in the registry."[39] This line of reasoning by the Examiner assumes that Kruger knows which resources should change as a result of the installation before the installation occurs, but Kruger explicitly does not know this information since there is no knowledge in Kruger of how the master computer was modified by the Examiner's presumed previous installations. In other words, the Examiner is taking the position that, since after the installation has occurred we know of a resource that has changed, and since such a resource can have an attribute that must change when

[39] *See* 09-16-2008 Advisory Action at page 2, "First" point.

the resource changes, then necessarily it was known before the installation occurred that this attribute should change. However, before the installation occurred, Kruger had no knowledge of which resources were going to change during the installation. Thus, the asserted dynamic attribute in Kruger cannot be considered a dynamic attribute as claimed, because the asserted dynamic attribute must necessarily change as soon as the resource is known (in retrospect) to be one that changed.

The failure of the Examiner's line of reasoning here becomes even more clear when considering the third recited element of claim 1: "identifying from the installation data the dynamic attribute that was not changed in the current software installation[.]" The Examiner has pointed to Column 8, lines 34-40 of Kruger as allegedly teaching this subject matter.[40] However, this portion of Kruger is describing his process of comparing the "after" state of the master computer to the "before" state of the same master computer to determine what resources and attributes have changed after the single described installation.[41] The claimed "identifying from the installation data the dynamic attribute that was not changed in the current software installation" cannot read on this portion of Kruger under the Examiner's own interpretation of the claim language.

In addressing this element of claim 1, the Examiner states that "a node contains information about a file such as the last modification date. When the difference calculator compares the node information between the nodes and determines that the information (e.g., the last modification date) did not change, it marks the node as 'same.' This means that the node did

---

[40] *See* 6-16-2008 Office Action at page 22-23.
[41] *See* Kruger at col. 7, line 25, to col. 9, line 39.

not change in the current software installation."[42] However this attribute (as the Examiner admits) has not changed because it's resource has not changed. Thus, this attribute cannot be the dynamic attribute under the Examiner's interpretation because that asserted attribute (last modification date) must have changed because it's resource changed under the Examiner's interpretation of previous claim elements. In other words, the Examiner is changing which resource (and thus which attribute) of Kruger he is reading the claim language on (switching from a first resource that changed to a second resource that didn't change during the single installation in Kruger) as he reads the claim language. This is a completely inappropriate approach to attempting to read a claim on cited art.

The language of claim 1 recites, "identifying from the installation data the dynamic attribute that was not changed in the current software installation[.]"[43] The dynamic attribute in this claim element is referring back to the "attribute that should have changed between the previous software installation and the current software installation [...]"[44] Thus, the Examiner is reading the same claimed attribute (which should have changed, but didn't) on two separate attributes in Kruger: a first modification date attribute that must change because it's resource changed, and a second modification date attribute that cannot change because it's resource didn't change. Thus, the rejection of claim 1 should be overturned for at least this additional reason.

Moreover, the installation data above is referring back to the data created "based at least in part on the comparison[.]"[45] The comparison is referring back to "generating a comparison of a current software installation, to a target computer, with a previous software installation, to the

---

[42] *See* 09-16-2008 Advisory Action at pages 2-3, "Third" point.
[43] Emphasis added.
[44] Emphasis added.
[45] Emphasis added.

same target computer, in a series of two or more software installations[.]"[46] Thus, in light of the arguments above, since Kruger is clearly describing the process of determining differences that result from a single installation of software to a single master computer, it is, strictly speaking, impossible for the claim language to read on Kruger, regardless of any combination with Breggin. Kruger is teaching a process of determining how a master computer has changed during an installation of software, so that this change information can be stored in a manifest for use in installing that same software to different computers.[47] Therefore, Kruger does not in any way teach or suggest, (emphasis added), "generating a comparison of a current software installation, to a target computer, with a previous software installation, to the same target computer, in a series of two or more software installations [...]; creating installation data for a resource, based at least in part on the comparison, the resource including attributes including a dynamic attribute and a static attribute, the dynamic attribute is an attribute that should have changed between the previous software installation and the current software installation, the static attribute is an attribute that should remain unchanged between the previous software installation and the current software installation; identifying from the installation data the dynamic attribute that was not changed in the current software installation", as recited in claim 1. Thus, for all of the above reasons, the rejection of claim 1 should be overturned.

B. Presenting Element of Claim is Not Found in the Art of Record

The Examiner acknowledges that Kruger does not disclose "presenting potential problems with the current software installation based on the identified dynamic attribute to

[46] Emphasis added.
[47] *See e.g.*, Kruger at col. 11, line 21, to col. 12, line 46.

facilitate verification of an installer for the software product development" and relies on Breggin for this missing feature.[48] Breggin, however, does not cure this deficiency of Kruger. Breggin explicitly states that, "An 'exception' is typically a difference between corresponding fields in the installation and installed databases or files."[49] Nothing in Breggin teaches or suggests presenting potential problems with a current software installation **based on an identified dynamic attribute**, where the dynamic attribute is an attribute that should have changed between the previous software installation and the current software installation, but that did not change, as expected.

The Examiner has cited to *In re Keller* in response to the this argument.[50] However, the argument being presented is not that prior art devices are not physically combinable; instead, the point being made is that the prior art references themselves do not teach or suggest the required limitation of claim 1. For example, the Examiner states that "Kruger discloses the identified dynamic attribute ... and Breggin is relied upon for its specific teaching of presenting potential problems with the current installation."[51] However, as noted above, Kruger fails to teach or suggests dynamic attributes, let alone identifying such dynamic attributes. Further, the proper test is "what the combined teaching of [the] references would have suggested to those of ordinary skill in the art."[52] In the present case, nothing in either Breggin or Kruger would suggest to those of ordinary skill in the art to flag as a potential problem an attribute that didn't change.

---

[48] *See* 06-16-2008 Office Action at page 5.
[49] *See* Breggin at col. 9, lines 58-60.
[50] *See* 11-28-2007 Office Action at page 23, lines 8-11.
[51] *See* 11-28-2007 Office Action at page 23, lines 22-24; and 6-16-2008 Office Action at page 24..
[52] *See* MPEP §2145, citing *In re Keller*.

The exception being reported in Breggin is an identified change, not the absence of a change.[53] As discussed above, Kruger creates a manifest showing all the changes made as the result of the installation, but explicitly discards (and does not report regarding) information about nodes that stayed the same.[54] Since Breggin's exception relates to a difference, the combined references do not teach or suggest presenting potential problems based on the absence of an expected difference. The Examiner fails to specifically address this point, and instead refers back the arguments based on Kruger for the dynamic attribute, and states that, "Breggin is relied upon for its specific teaching of presenting potential problems with the current installation."[55] Thus, the Examiner has acknowledged with respect to claim 1 that Breggin does not teach a dynamic attribute as claimed. The Examiner has also asserted that the identified dynamic attribute that did not change in Kruger is for a node marked as "same."[56] But Kruger explicitly teaches such nodes are discarded.[57] This is because such nodes do not represent potential problems with the installation in Kruger. Thus, there is no reason to combine Kruger and Breggin as suggested by the Examiner, and the claimed feature relating to presenting potential problems based on the absence of an expected difference ("presenting potential problems with the current software installation based on the identified dynamic attribute to facilitate verification of an installer for the software product development", as recited in claim 1) is nowhere to be found in either Breggin or Kruger.

Therefore, the proposed Kruger-Breggin combination does not teach or suggest each and every limitation of claim 1, and claim 1 should be in condition for allowance. Independent

---

[53] *See e.g.*, Breggin at col. 9, lines 58-60.
[54] *See e.g.*, Kruger at col. 8, lines 12-48.
[55] *See* 6-16-2008 Office Action at page 24; and 09-16-2008 Advisory Action at page 3, "Sixth" point.
[56] *See* 09-16-2008 Advisory Action at pages 2-3, "Third" point.
[57] *See e.g.*, Kruger at col. 8, lines 33-48.

claims 10 and 25 recite similar limitations as claim 1; thus, these claims should be in condition for allowance for at least similar reasons as discussed above. Claims 2, 5-9, 12, 14-18, 26 and 28-32 depend generally from claim 1, 10, or 25, and are allowable for at least the reasons provided above.

Thus, for all of the above reasons, Ground of Rejection I, based on Kruger in view of Breggin, should be overturned.

II. Rejection based on Breggin in view of Kruger: Neither of these references, either alone or in combination, describe the subject matter of claims 19, 20, 22 and 23.

Independent claim 19 recites, "a build controller; an install controller comprising a database including a baseline recording expectations of a dynamic attribute and a static attribute for one or more resources associated with a software installer, the dynamic attribute is an attribute that should have changed between a previous software installation and a current software installation, the static attribute is an attribute that should remain unchanged between the previous software installation and the current software installation; and one or more install slave machines; wherein the build controller automatically triggers the install controller to initiate installer tests as part of a software build process, and the install controller automatically dispatches installation to the one or more install slave machines and collects test results to be presented in a report comprising a baseline-update interface."[58] The Examiner acknowledges that Breggin fails to disclose "a dynamic attribute and a static attribute for one or more resources

[58] Emphasis added.

associated with a software installer" and relies on Kruger to cure the deficiencies of Breggin.[59] However, as addressed above in connection with Ground of Rejection I, Kruger also does not teach or suggest "a dynamic attribute and a static attribute for one or more resources associated with a software installer" as recited in claim 19. Therefore, the suggested Breggin-Kruger combination does not teach or suggest all the elements of claim 19, and the rejection of claim 19 should be overturned for at least this reason. Claims 20, 22 and 23 depend generally from claim 19 and are allowable for at least the reasons provided above. Thus, for all of the above reasons, Ground of Rejection II, based on Breggin in view of Kruger, should be overturned.

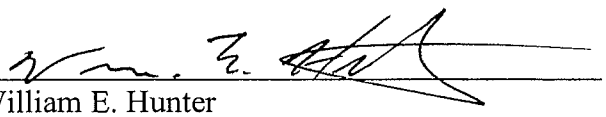
III. Rejection based on Breggin in view of Kruger and Suorsa: None of these references, either alone or in combination, describe the subject matter of claim 21.

Claim 21 depends from claim 19. Since Suorsa does not cure the noted deficiencies of Breggin and Kruger, claim 21 should be allowable for at least the reasons provided above. Thus, for all of the above reasons, Ground of Rejection III, based on Breggin in view of Kruger and Suorsa, should be overturned.

Please apply the $540 brief fee and any other necessary charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: Nov. 17, 2008



William E. Hunter
Reg. No. 47,671

Fish & Richardson P.C.
PTO Customer No. 21876
Telephone: (858) 678-5070
Facsimile: (877) 769-7945
10871488.doc

[59] *See* 6-16-2008 Office Action at page 13.

Appendix of Claims

1. A machine-implemented method comprising:

generating a comparison of a current software installation, to a target computer, with a previous software installation, to the same target computer, in a series of two or more software installations during a software product development;

creating installation data for a resource, based at least in part on the comparison, the resource including attributes including a dynamic attribute and a static attribute, the dynamic attribute is an attribute that should have changed between the previous software installation and the current software installation, the static attribute is an attribute that should remain unchanged between the previous software installation and the current software installation;

identifying from the installation data the dynamic attribute that was not changed in the current software installation; and

presenting potential problems with the current software installation based on the identified dynamic attribute to facilitate verification of an installer for the software product development.

2. The machine-implemented method of claim 1, further comprising:

identifying from the installation data the static attribute that was changed in the current software installation; and

presenting potential problems with the current software installation based on the identified static attribute to facilitate verification of the installer for the software product development.

5. The machine-implemented method of claim 1, further comprising tracking expectations for the resource in a primary installation baseline and a secondary installation baseline, and wherein presenting the potential problems comprises presenting a baseline-update interface by transmitting markup language data.

6. The machine-implemented method of claim 1, further comprising excluding a set of resources from the generated comparison for the software product development.

7. The machine-implemented method of claim 5, wherein expectations of resource changes, including the installation data, are stored in a relational database indexed by date, platform, language, and product configuration.

8. The machine-implemented method of claim 1, wherein the attributes comprising modification date stamp information, file size information, security permissions information, and checksum information.

9. The machine-implemented method of claim 1, wherein the resource comprises a file and a system registry, and the installation data comprises deletion, addition, and modification of the resource.

10. A software product tangibly embodied in a machine-readable storage device, the software product comprising instructions operable to cause one or more data processing apparatus to perform operations comprising:

generating a comparison of a current software installation, to a target computer, with a previous software installation, to the same target computer, in a series of two or more software installations during a software product development;

creating installation data for a resource, based at least in part on the comparison, the resource including attributes including a dynamic attribute and a static attribute, the dynamic attribute is an attribute that should have changed between the previous software installation and the current software installation, the static attribute is an attribute that should remain unchanged between the previous software installation and the current software installation;

identifying from the installation data the dynamic attribute that was not changed in the current software installation; and

presenting potential problems with the current software installation based on the identified dynamic attribute to facilitate verification of an installer for the software product development.

12. The software product of claim 10, wherein the operations further comprise

identifying from the installation data the static attribute that was changed in the current software installation; and

presenting potential problems with the current software installation based on the identified static attribute to facilitate verification of the installer for the software product development.

14. The software product of claim 12, wherein the operations further comprise tracking expectations of resource changes in a primary installation baseline and a secondary installation baseline, and wherein presenting the potential problems comprises presenting a baseline-update interface by transmitting markup language data.

15. The software product of claim 12, wherein the operations further comprise excluding a set of resources from the generated comparison for the software product development.

16. The software product of claim 12, wherein expectations of resource changes are stored in a relational database indexed by date, platform, language, and product configuration.

17. The software product of claim 10, wherein the attributes comprising modification date stamp information, file size information, security permissions information, and checksum information.

18. The software product of claim 12, wherein the resource comprises a file and a system registry, and the installation data comprises deletion, addition, and modification of the resource.

19. A system comprising:

a build controller;

an install controller comprising a database including a baseline recording expectations of a dynamic attribute and a static attribute for one or more resources associated with a software installer, the dynamic attribute is an attribute that should have changed between a previous software installation and a current software installation, the static attribute is an attribute that should remain unchanged between the previous software installation and the current software installation; and

one or more install slave machines;

wherein the build controller automatically triggers the install controller to initiate installer tests as part of a software build process, and the install controller automatically dispatches installation to the one or more install slave machines and collects test results to be presented in a report comprising a baseline-update interface.

20. The system of claim 19, wherein the one or more install slave machines comprise multiple computers.

21. The system of claim 19, wherein the install controller communicates with the one or more install slave machines using Simple Object Access Protocol.

22. The system of claim 19, wherein the baseline-update interface comprises a web-based user interface allowing baseline updates across SKU, language, operating system, and custom/non-custom installs, in combination or all at once.

23. The system of claim 19, wherein the attributes comprising modification date stamp information, file size information, security permissions information, and checksum information.

25. A system comprising:

a user interface device; and

one or more computers operable to interact with the user interface device and to perform operations comprising:

generating a comparison of a current software installation, to a target computer, with a previous software installation, to the same target computer, in a series of two or more software installations during a software product development;

creating installation data for a resource, based at least in part on the comparison, the resource including attributes including a dynamic attribute and a static attribute, the dynamic attribute is an attribute that should have changed between the previous software installation and the current software installation, the static attribute is an attribute that should remain unchanged between the previous software installation and the current software installation;

identifying from the installation data the dynamic attribute that was not changed in the current software installation; and

presenting potential problems with the current software installation based on the identified dynamic attribute to facilitate verification of an installer for the software product development.

26. The system of claim 25, wherein the operations further comprise:

identifying from the installation data the static attribute that was changed in the current software installation; and

presenting potential problems with the current software installation based on the identified static attribute to facilitate verification of the installer for the software product development.

28. The system of claim 25, wherein the operations further comprise tracking expectations for the resource in a primary installation baseline and a secondary installation baseline, and wherein presenting the potential problems comprises presenting a baseline-update interface by transmitting markup language data.

29. The system of claim 25, wherein the operations further comprise excluding a set of resources from the generated comparison for the software product development.

30. The system of claim 28, wherein expectations of resource changes, including the installation data, are stored in a relational database indexed by date, platform, language, and product configuration.

31. The system of claim 25, wherein the attributes comprise modification date stamp information, file size information, security permissions information, and checksum information.

32. The system of claim 25, wherein the resource comprises a file and a system registry, and the installation data comprises deletion, addition, and modification of the resource.

Evidence Appendix

None.

Related Proceedings Appendix

None.